Měřič polétavého prachu s PMS 7003 verze 2.0

V tomto případě udělám výjimku a zveřejním na svém webu návod pro měřič polétavého prachu – tedy čistoty ovzduší, za kterým stojí kolegium autorů. Já jsem jen zasel pomyslné semínko svým původním velmi jednoduchým řešením na odečet hodnot ze senzoru PMS 7003. Bohužel z časových důvodů, ale hlavně nedostatečné erudovanosti na poli programování Arduina jsem výsledek přenechal odborníkům.

Zde je odkaz na můj původní návod „hloupého“ měřiče, který jen odečítal a zobrazoval hodnoty ze senzoru v reálném čase.

O co jde?

Měřič polétavého prachu je Opens Source projekt pro nekomerční použití, pomocí kterého by si každý, řekněme kutil, či nadšenec měl být schopen sestavit přístroj za pár dolarů, který bude umět změřit míru znečištění ovzduší na základě měření prachových částic různých velikostí. Vyspělejší verze 2.0 pracuje i s datalogerem na SD kartu, teploměrem a vlhkoměrem. Pokud vám není lhostejno co dýcháte, obzvláště na vesnici, kdy soused přiloží kde čím a chcete své stížnosti, či připomínky podložit reálným měřením, nebo jen z akademických důvodů si chcete udělat obrázek o tom co dýcháte a jaký to má vliv na váš zdravotní stav, zde je návod pro vás.

Verze

- PMS 7003 – 2.0 (21.12.2020) nyní je hotová první verze „chytřejšího“ snímače. Určitě bude nutno odladit nějaké nedostatky. Dojde-li ke změnám, budou zveřejněny zde.

Program

Zde přikládám program pro Arduino MEGA, jehož autorem je Milan Mačuga. Pro pohodlnou komunikaci s čidlem, SD kartou a teploměrem se muselo přejít na Arduino MEGA, neboť UNO má již pro tyto účely malou paměť. Kod je velmi krásně okomentován, není asi třeba k tomu moc dodávat

```
// Toto je:

char verzeSouboru = "05b_15_11_2020_PMS.INO";

/*   - oprava: 14.11.2020 - proměnná value[] změněna z byte na int, načítala
se hodnota z PMS jen do 255.
     úsek (Zobrazení na LCD) - změna pozic více doleva pro zobrazení hodnot
nad 999, zhuštění hodnot na displeji
     + výmaz historie verzí

     Předchozí:
     05_12_11_2020_PMS.INO
     Verze, nahraná v předaném díle


  // ********* DOKUMENTACE ***************
   Čidlo měření kvality ovzduší
    (c) Milan Mačuga, mm.online@seznam.cz, vytvořeno 11/2020

    Zapojení:
    ========================================
    Čidlo: Plantower PMS7003 -------:
    Tx čidla PMS zapojen na pin 17 MEGA = serial 2
 *   *  (#define pmsSerial Serial2 // green wire from PMS5003 goes to pin 17
on **Mega**)
     u UNO se použije SoftwareSerial a použije se pak libovolný pin
     -------
     SDA/SCL pro MEGA SDA=20 SCL=21 = I2C
     --------
     Displej - SDA/SCL
     SD karta - MISO-50   MOSI-51   SCK-52  CS/SDCS-53 na MEGA
     Klávesnice klávesa "1" = pin D3  pak D4 - D5 - D6 a společný D7 pro SW
zem pomocí LOW
     RTC - SDA/SCL
     Vlhkoměr/teploměr typ AM2320 - SDA/SCL
     PMS - D17 (Serial2 u MEGA), použit pouze pin TX, VCC a GND

*/
```

```
// ********* PRG **************
// Zadání konstanty pro ukládání dat
unsigned long interval = 60000; // v milisekundách = 1 minuta
unsigned long previousMillis = 0; // proměnné pro překlápění
// currentMillis je definovan níže v části ukládání na SD
//*********************************

/* Aktuální datum a čas se do PMS vždy nahraje při downloadu programu
    Letní čas není vyřešen, musí se doprogramovat
    Klávesa 1 slouží ke zhasínání displeje. Klávesy 2-4 mají sloužit k
prohlížení dat
    tj. nahoru, dolů a MENU (Enter)
*/

// Setup hardware
// Membránová klávesnice 1x4 - pole s piny připojených tlačítek
const int tlacitka[] = {4, 3, 6, 5}; // piny s tlačítky
const int tlacitkaGND = 7; // piny s tlačítky
bool zhasniLCD = HIGH; // proměnná pro zjištění stavu podsvícení
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

// proměnná stavu tlačítka
int stisk = 0;

// Teploměr a vlhkoměr AM2320
#include <AM2320.h>

// inicializace modulu z knihovny
AM2320 senzor;
float aktTeplota; // proměnná pro předávání
float aktVlhkost; // -/-

//#include <SoftwareSerial.h> // pro UNO, nano ...

#include <LiquidCrystal_I2C.h> // Library for LCD
// Wiring: SDA pin is connected to A4 and SCL pin to A5.
// Connect to LCD via I2C, default address 0x27 (A0-A2 not jumpered)
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x21, 20, 4); // Change to
```

```
(0x27,16,2) for 16x2 LCD.
// Aktuálně použitý displej má adresu 21, při výměně displeje nutno změnit
dle faktické adresy
// nebo propojit na desce a přeadresovat pájením na 21

// připojení potřebných knihoven
#include <Wire.h>

// RTC Hodiny reálného času DS3231
#include <RtcDS3231.h>
RtcDS3231<TwoWire> Rtc(Wire); // SDA-SCL ! - Wire.h musí být před tímto !
char datestring[20]; // pro proceduru printDateTime(const RtcDateTime& dt) -
předává čas ve stringu

// SD krta
#include <SD.h>
const int chipSelect = 53;

// PMS čidlo - definice připojení - jsou 2 možnosti, přímo na Serial 2 (MEGA)
nebo Software Serial (UNO)

//SoftwareSerial pmsSerial(3, 2); //RX/TX - používá se v **UNO, nano...**

/* Poznámky k možnosti, nacpat to do UNO:
   Problém je, že do UNO se to s RTC a SD knihovnou prostě nevleze...
   Software Serial jde použít pouze bez SD knihovny, pak už je to velké...

   Ukázka při pokusu nahrát 02b do UNA:

  Projekt zabírá 14832 bytů (45%)  úložného místa pro program. Maximum je
32256 bytů.
  data section exceeds available space in board

  Globální proměnné zabírají 2056 bytů (100%)  dynamické paměti, -8 bytů
zůstává pro lokální proměnné.
  Maximum je 2048 bytů.
  Nedostatek peměti; na http://www.arduino.cc/en/Guide/Troubleshooting#size
naleznete typy jak velikost redukovat.
  Nastala chyba při kompilaci u desky Arduino Uno.
```

```
*/

#define pmsSerial Serial2 // green wire from PMS5003 goes to pin 17 on
**Mega**
// purple goes to 5V
// orange goes to negative

int value[13]; // pro načítání proměnných z PMS
//const int ledPin =  8;      // číslo pinu LED
//bool ledState;



void setup() {

  // inicializace pole tlačítek
  for (int x = 0; x < 4; x++)
  {
    // zapojení tlačítek 1-4 jako vstup s pull-up odporem
    pinMode(tlacitka[x], INPUT_PULLUP);
  }
  // Společný vodič je napojen na pin7 a ten dělá GND
  pinMode(tlacitkaGND, OUTPUT);
  digitalWrite(tlacitkaGND, LOW);

  Serial.begin(9600);

  Serial.print("Verze souboru: "); Serial.println(verzeSouboru);
  // LCD setup
  lcd.backlight();
  Serial.println("LCD Backlight()");
  lcd.begin();
  Serial.println("LCD Begin()");
  lcd.clear();
  Serial.print("Initializing SD card...");

  lcd.setCursor(0, 0); lcd.print("Verze");
  lcd.setCursor(0, 1); lcd.print(verzeSouboru);
  lcd.setCursor(0, 2); lcd.print(".ino  ");
```

```
  delay(5000);
  lcd.clear();

  if (!SD.begin(chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("1. is a card inserted?");
    Serial.println("2. is your wiring correct?");
    Serial.println("3. did you change the chipSelect pin to match your shield
or module?");
    Serial.println("Note: press reset or reopen this serial monitor after
fixing your issue!");
    lcd.setCursor(0, 0); lcd.print("Chybicka...  ");
    lcd.setCursor(0, 1); lcd.print("nemas tam SD kartu");
    lcd.setCursor(0, 2); lcd.print("nebo je poskozena  ");
    lcd.setCursor(0, 3); lcd.print("Vypni! a vloz kartu");
    while (true);
  }

  Serial.println("initialization SD done.");
  lcd.clear();
  lcd.setCursor(13, 2);
  lcd.print("SDwr_OK"); // zapisuju s předstihem, protože jinak to tam bude
až za (interval)...
  //lcd.noBacklight(); // vypínání displeje
  delay(2000);

  // sensor PMS baud rate is 9600 !
  pmsSerial.begin(9600);
  //pinMode(ledPin, OUTPUT);
  Serial.println("za PMS Serial");
  Serial.print("compiled: ");
  Serial.print(__DATE__);
  Serial.println(__TIME__);

  Rtc.Begin();
  RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__); // nastavení
systémového času z PC

  if (!Rtc.IsDateTimeValid())
```

```
  {
    if (Rtc.LastError() != 0)
    {
      // we have a communications error
      // see https://www.arduino.cc/en/Reference/WireEndTransmission for
      // what the number means
      lcd.setCursor(0, 13); lcd.print("RTC !!");
      delay(5000);

      Serial.print("RTC communications error = ");
      Serial.println(Rtc.LastError());
    }
    else
    {
      // Common Causes:
      //    1) first time you ran and the device wasn't running yet
      //    2) the battery on the device is low or even missing

      Serial.println("RTC lost confidence in the DateTime!");
      lcd.clear();
      lcd.setCursor(0, 0);  lcd.print("Vymen ihned baterii ");
      lcd.setCursor(0, 1);  lcd.print("v RTC modulu !!     ");
      lcd.setCursor(0, 2);  lcd.print("...Nemam cas :o) ");
      delay(8000);
      lcd.clear();

      // following line sets the RTC to the date & time this sketch was
compiled
      // it will also reset the valid flag internally unless the Rtc device
is
      // having an issue

      Rtc.SetDateTime(compiled);
    }
  }

  if (!Rtc.GetIsRunning())
  {
    Serial.println("RTC was not actively running, starting now");
```

```
    Rtc.SetIsRunning(true);
  }

  RtcDateTime now = Rtc.GetDateTime();
  if (now < compiled)
  {
    Serial.println("RTC is older than compile time!  (Updating DateTime)");
    Rtc.SetDateTime(compiled);
  }
  else if (now > compiled)
  {
    Serial.println("RTC is newer than compile time. (this is expected)");
  }
  else if (now == compiled)
  {
    Serial.println("RTC is the same as compile time! (not expected but all is
fine)");
  }

  // never assume the Rtc was last configured by you, so
  // just clear them to your needed state
  Rtc.Enable32kHzPin(false);
  Rtc.SetSquareWavePin(DS3231SquareWavePin_ModeNone);

  //
  Serial.println("--- Konec Setup(), vše proběhlo úspěšně a jde se do
Loop()");
  lcd.setCursor(0, 0);  lcd.print("Cekam na");
  lcd.setCursor(0, 1);  lcd.print("data z  ");
  lcd.setCursor(0, 2);  lcd.print("cidla   ");
  delay(3000);

}

struct pms5003data {
  uint16_t framelen;
  uint16_t pm10_standard, pm25_standard, pm100_standard;
  uint16_t pm10_env, pm25_env, pm100_env;
  uint16_t particles_03um, particles_05um, particles_10um, particles_25um,
```

```
particles_50um, particles_100um;
  uint16_t unused;
  uint16_t checksum;
};

/*  Toto obsahují proměnné
    -> ---------------------------------------
  13:37:39.335 -> Concentration Units (standard)
  13:37:39.368 -> PM 1.0: 11    PM 2.5: 12    PM 10: 13
  13:37:39.402 -> ---------------------------------------
  13:37:39.436 -> Concentration Units (environmental)
  13:37:39.470 -> PM 1.0: 11    PM 2.5: 12    PM 10: 13
  13:37:39.505 -> ---------------------------------------
  13:37:39.573 -> Particles > 0.3um / 0.1L air:1818
  13:37:39.609 -> Particles > 0.5um / 0.1L air:590
  13:37:39.642 -> Particles > 1.0um / 0.1L air:53
  13:37:39.675 -> Particles > 2.5um / 0.1L air:1
  13:37:39.709 -> Particles > 5.0um / 0.1L air:1
  13:37:39.747 -> Particles > 10.0 um / 0.1L air:0
  13:37:39.747 -> ---------------------------------------

*/

struct pms5003data data;

void loop() {
  // Resetni se každý den o tom čase (nebo po 24 po spuštění)

  checkKBD1(); // zapnutí nebo vypnutí displeje, později i další funkce
tlačítek

  // Načtení PMS;

  if (readPMSdata(&pmsSerial)) { // reading data was successful!
    // rtcCheck(); // odkomentuj jen pro ucely ladeni

    // Načtení aktuálního data a času do proměnné
    RtcDateTime now = Rtc.GetDateTime();
    setDateString(now); // naplnění datestringu správnými dvoucifernými
```

```
hodnotami
    nactiTeplotu(); // a ulož je do aktTeplota a aktVlhkost

    // Zobrazení procesu na terminálu
    Serial.println();
    Serial.println("--- Úspěšně načteno ------");
    Serial.println("Concentration Units (standard)");
    Serial.print("PM 1.0: "); Serial.print(value[0]);
    Serial.print("\t\tPM 2.5: "); Serial.print(value[1]);
    Serial.print("\t\tPM 10: "); Serial.println(value[2]);
    Serial.print("Teplota: "); Serial.print(aktTeplota); Serial.print(" stC,
vlhkost: "); Serial.print(aktVlhkost); Serial.println("%");
    Serial.println("---------------------------------------");

    // Zobrazení na LCD
    lcd.setCursor(6, 0); lcd.print(value[0]); lcd.print("  ");
    lcd.setCursor(6, 1); lcd.print(value[1]); lcd.print("  ");
    lcd.setCursor(6, 2); lcd.print(value[2]); lcd.print("  ");
    /* původní, vymazat po odladění z kódu
      lcd.setCursor(9, 0); lcd.print(value[0]); lcd.print("  ");
      lcd.setCursor(9, 1); lcd.print(value[1]); lcd.print("  ");
      lcd.setCursor(9, 2); lcd.print(value[2]); lcd.print("  ");
    */

    lcd.setCursor(13, 0);  lcd.print(aktTeplota, 1); lcd.print("  ");
    lcd.setCursor(18, 0);  lcd.print((char) 223); lcd.print("C"); // char
223=snak stupne
    lcd.setCursor(13, 1);  lcd.print(aktVlhkost, 1); lcd.print(" ");
    lcd.setCursor(18, 1);  lcd.print("%H");

    lcd.setCursor(0, 0);  lcd.print("PM1.0=");
    lcd.setCursor(0, 1);  lcd.print("PM2.5=");
    lcd.setCursor(0, 2);  lcd.print("PM10 =");
    /* původní, vymazat po odladění z kódu
      lcd.setCursor(0, 0);  lcd.print("PM 1.0 =");
      lcd.setCursor(0, 1);  lcd.print("PM 2.5 =");
      lcd.setCursor(0, 2);  lcd.print("PM 10  =");
    */
    lcd.setCursor(0, 3);   lcd.print(datestring); // zobraz datum a čas z
```

```
proměnné



    // naplň data z PMS čidla
    value[0] = (data.pm10_standard);
    value[1] = (data.pm25_standard);
    value[2] = (data.pm100_standard);

    // Příprava dat pro zápis na SD kartu
    String dataProZapis = datestring; dataProZapis += ","; // naplň datum a
čas
    dataProZapis = dataProZapis + String(value[0]); dataProZapis += ",";
//pm10_standard
    dataProZapis = dataProZapis + String(value[1]); dataProZapis += ",";
//pm25_standard
    dataProZapis = dataProZapis + String(value[2]); dataProZapis += ",";
//pm100_standard
    dataProZapis = dataProZapis + String(aktTeplota); dataProZapis += ",";
    dataProZapis = dataProZapis + String(aktVlhkost); dataProZapis += ",";
    // Konec plnění

    // SD karta - zápis
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis > interval) {

      File dataFile = SD.open("datalog.txt", FILE_WRITE);
      // if the file is available, write to it:
      if (dataFile) {
        dataFile.println(dataProZapis);
        dataFile.close();
        // print to the serial port too:
        Serial.println("Zapsano na SD...");
        lcd.setCursor(13, 2);
        lcd.print("SDwr_OK");
        previousMillis = currentMillis; // Zápis byl úspěšný, nuluju a čekám
další minutu
      }
```

```
      // if the file isn't open, pop up an error:
      else {
        Serial.println("error opening datalog.txt");
        lcd.setCursor(13, 2);
        lcd.print("SD -!!-"); // POZOR, tři vykřičníky blokují zavedení
programu...
      }  // Zápis na SD kartu se nepovedl

    } // už je čas to zapsat :o) - ale čekej až na úspěšný zápis na SD ***


  } // readPMSdata
  //delay(100); // POZOR pokud se tady dá vyšší hodnota, bude CheckSum
Failure! i bez delay vše jede, jak má.
  // delay jsem odstranil, protože už to házelo checksum = velké zpoždění při
průchodu loop()

}

// Kontrola RTC - pro DEBUG
void rtcCheck() {
  if (!Rtc.IsDateTimeValid())
  {
    if (Rtc.LastError() != 0)
    {
      // we have a communications error
      // see https://www.arduino.cc/en/Reference/WireEndTransmission for
      // what the number means
      Serial.print("RTC communications error = ");
      Serial.println(Rtc.LastError());
    }
    else
    {
      // Common Causes:
      //    1) the battery on the device is low or even missing and the power
line was disconnected
      Serial.println("RTC lost confidence in the DateTime!");
    }
  }
```

```
} // RTC Check

void nactiTeplotu() {
  switch (senzor.Read()) {
    // v případě stavu "2" je chyba komunikace
    case 2:
      Serial.println("Chybny CRC soucet, chyba v komunikaci!");
      break;
    // v případě stav "1" je senzor offline nebo špatně připojen
    case 1:
      Serial.println("Senzor offline!");
      break;
    // v případě stavu "0" je vše v pořádku
    // a můžeme vytisknout údaje
    case 0:
      float teplota = senzor.t;
      float vlhkost = senzor.h;
      aktTeplota = teplota;
      aktVlhkost = vlhkost;
      break;
  }
}// NactiTeplotu

void checkKBD1() { // reakce na stisk tlačítka "1"
  // read the pushbutton input pin:
  buttonState = digitalRead(tlacitka[0]);
  //Serial.println(zhasniLCD);

  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      zhasniLCD = !zhasniLCD; // když stisknuto, změň stav
      delay(50); //pro dokmit
    }
    // save the current state as the last state, for next time through the
loop
    lastButtonState = buttonState;
```

```
    // změna stavu displeje po stisku
    if (zhasniLCD == HIGH) {
      lcd.noBacklight();
    }
    else
    {
      lcd.backlight();
    }

  }
} // checkKBD1


//RTC správné zobrazení času se dvěma 00
#define countof(a) (sizeof(a) / sizeof(a[0]))
void setDateString(const RtcDateTime& dt)
{
  //    char datestring[20];

  snprintf_P(datestring,
             countof(datestring),
             PSTR("%02u/%02u/%04u %02u:%02u:%02u"),
             dt.Day(),
             dt.Month(),
             dt.Year(),
             dt.Hour(),
             dt.Minute(),
             dt.Second() );
}

boolean readPMSdata(Stream *s) {
  if (! s->available()) {
    return false;
  }

  // Read a byte at a time until we get to the special '0x42' start-byte
  if (s->peek() != 0x42) {
    s->read();
    return false;
```

```
  }

  // Now read all 32 bytes
  if (s->available() < 32) {
    return false;
  }

  uint8_t buffer[32];
  uint16_t sum = 0;
  s->readBytes(buffer, 32);

  // get checksum ready
  for (uint8_t i = 0; i < 30; i++) {
    sum += buffer[i];
  }

  /* debugging
    for (uint8_t i=2; i<32; i++) {
    Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
    }
    Serial.println();
  */

  // The data comes in endian'd, this solves it so it works on all platforms
  uint16_t buffer_u16[15];
  for (uint8_t i = 0; i < 15; i++) {
    buffer_u16[i] = buffer[2 + i * 2 + 1];
    buffer_u16[i] += (buffer[2 + i * 2] << 8);
  }

  // put it into a nice struct :)
  memcpy((void *)&data, (void *)buffer_u16, 30);

  if (sum != data.checksum) {
    Serial.println("Checksum failure");
    return false;
  }
  // success!
  return true;
```

```
} // Čtení PMS dat - end
```

Hardware

K vylepšené verzi projektu je použito pár běžných komponent a celek je zasazen do krabičky na míru navržené a vyrobené pomocí 3D tisku. Použitý RTC modul nabíjí baterii, proto je nutno použít nabíjecí knoflíkovou baterii LIR2032

- ARDUINO MEGA
- 20×4 LCD displej 2004 zelený + I2C převodník
- I2C teploměr a vlhkoměr DHT12 AM2320 digitální
- Mini RTC Hodiny reálného času DS1307
- Senzor prachových částic PMS 7003
- Membránová klávesnice 1×4 maticová

Tištěná krabička

Pro celý aparát byla navržena 3D tištěná krabička, rovněž panem Milanem Mačugou

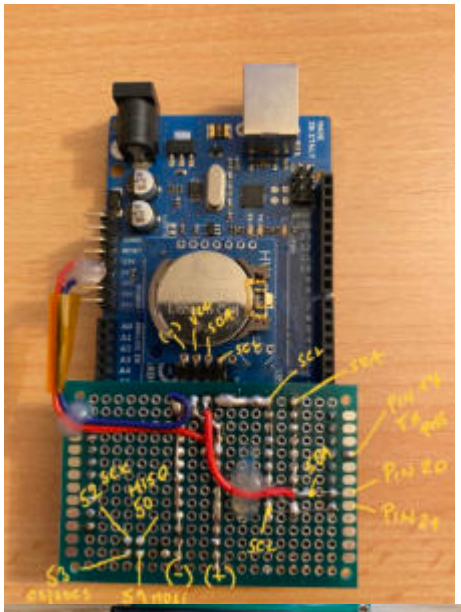
>>STL soubory pro tisk krabičky ke stažení zde<<



Sestavení

Schéma zapojení zatím není k dispozici, ale drtivá většina periferie je připojena přes I2C sběrnici, případně SPI, propojení je zakresleno ve fotografiích, případně okomentováno ve zdrojovém kódu. Nic složitého.

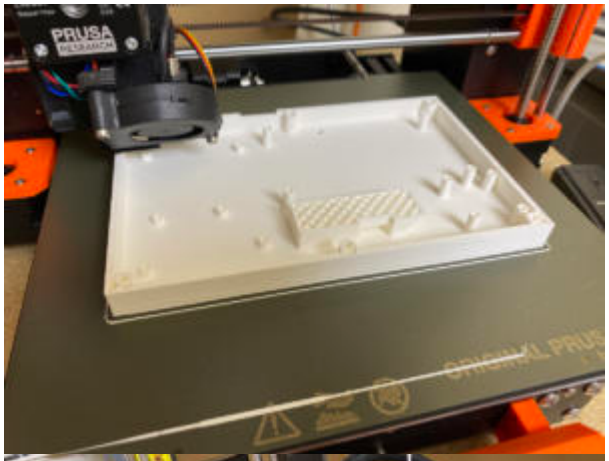
Měřič polétavého prachu s PMS 7003 verze 2.0



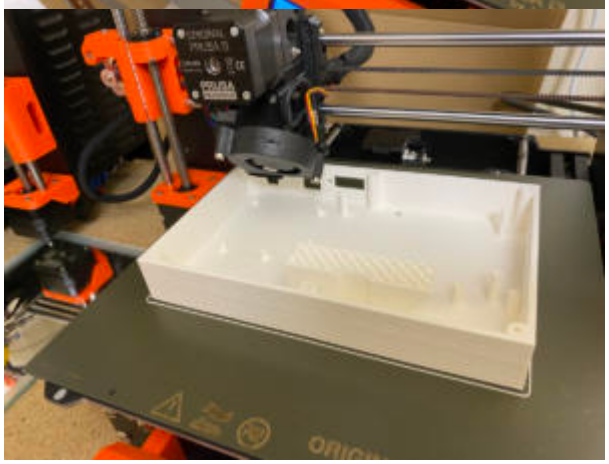




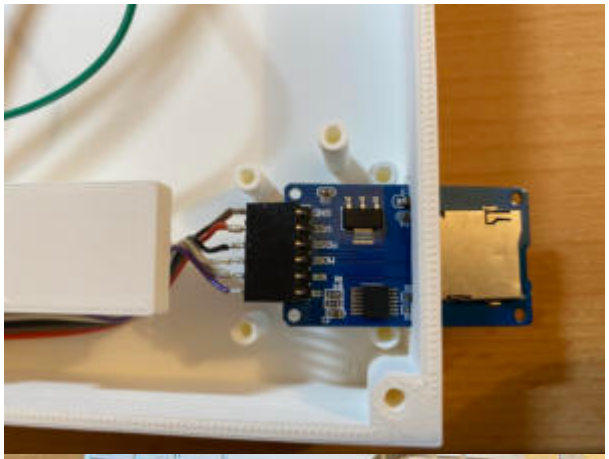
Měřič polétavého prachu s PMS 7003 verze 2.0

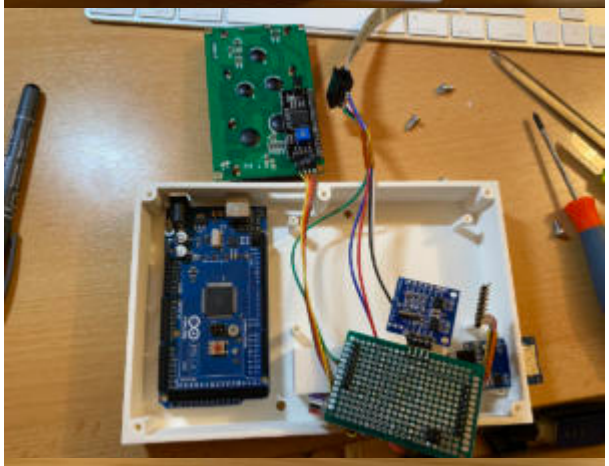






Měřič polétavého prachu s PMS 7003 verze 2.0

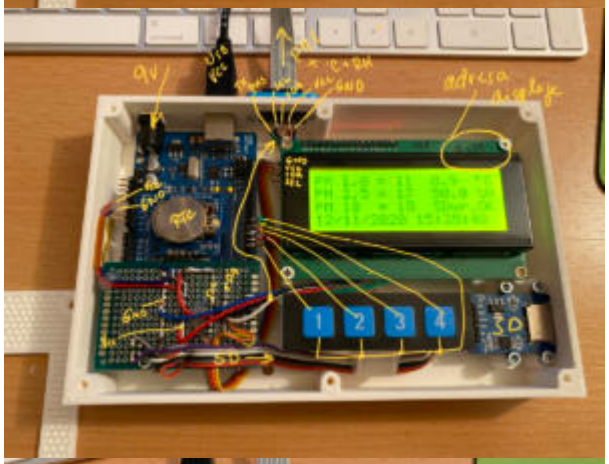






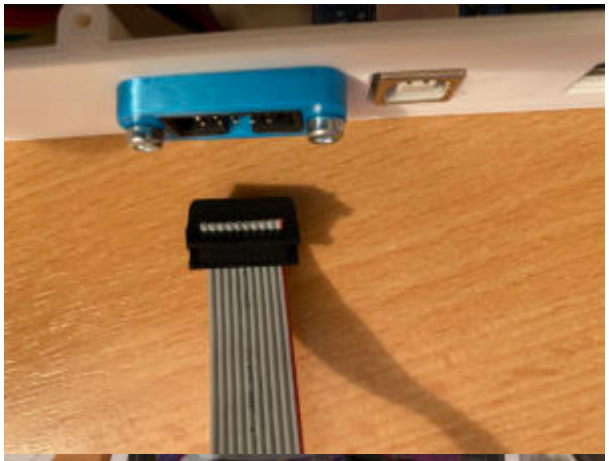
Měřič polétavého prachu s PMS 7003 verze 2.0





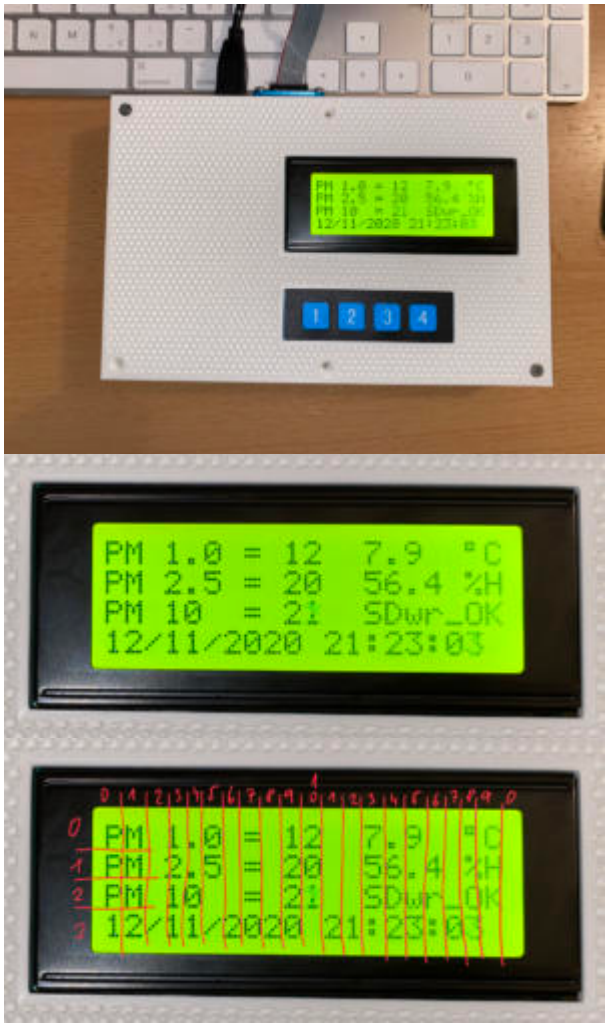


Měřič polétavého prachu s PMS 7003 verze 2.0









Závěr

Projekt je stále ve vývoji, možná se dočká i nějaké aplikace pro vyhodnocování naměřených dat atd. V současné chvíli se pracuje na kalibraci zařízení (porovnáním s měřičem ČHMI) ovšem jaksi bez certifikace, která stojí skoro jako celá výplata. Pro tyto free účely tedy v naprosto šílených finančních relacích.

V případě zájmu, či podrobnější informace k tomuto projektu nás neváhejte kontaktovat pavel.kopp(zavináč)seznam.cz

Měřič polétavého prachu s PMS 7003 verze 2.0

Autoři:

- Milan Mačuga
- Pavel Kopp
- Jan Kondziolka

Total Page Visits: 2951 - Today Page Visits: 3